14 | 性能分析概述：性能问题归根结底是什么原因？

2019-12-27 庄振运

性能工程高手课 进入课程 >

讲述：庄振运
时长 11:47　大小 10.80M

你好，我是庄振运。

前面几讲，我们讨论了有关性能测试的内容，了解了各种测试的种类、测试的规划、测试的工具以及执行测试的经验教训。在整个性能优化的世界里，性能测试是基础，在这一基础上，我们才能对各种性能问题进行性能分析，并进行相应的性能优化。

从这一讲开始，我们来探讨一下常见的各种性能问题，和如何进行相应的性能分析。我们首先宏观地介绍一下性能的外部指标、内部瓶颈和资源制约，以及如何正确地进行性能分析。然后接下来的几讲分别重点讨论 CPU、内存、存储和网络几个领域。

性能的外部指标

谈论应用程序系统和互联网服务的性能时，我们首先要清楚外部的性能指标是什么。最重要也最普遍的指标有三个：服务延迟（Service Latency）、吞吐率（Throughput）和资源使用率（Resource Utilization）。我们分别来看看。

服务延迟

服务延迟（或者叫访问延迟），指的是客户发出的请求被成功服务的时间。

虽然具体的定义和度量有很多种，比如有些度量只考虑系统本身的服务时间，而不考虑其它因素（比如网络传输时间）。但我个人坚持，这个指标是直接针对客户体验的，因此不能仅仅从自己的系统角度衡量，而必须是端到端的延迟度量（End To End, or E2E）。因为只有这样来定义这一指标，才能准确地抓住“外部性能”这一特点。

任何系统和服务的设计和搭建都是为了给客户来用的，如果不紧紧抓住“客户体验”这一根本目标，性能测试、性能分析，以至于性能优化也就成了无的放矢，基本失去了意义。

吞吐率

吞吐率指的是单位时间（比如每秒钟）可以成功处理的请求数或任务数。

这一指标和前面讲的访问延迟指标相辅相成，一个注重时间，就是服务延迟；一个注重空间，也就是系统容量。一个系统的外部性能主要受到这两个条件的约束，缺一不可。

比如，一个在线聊天服务系统，可以提供每秒钟一百万的吞吐率，但是客户的访问延迟是 5 分钟以上，那么这个“一百万的吞吐率”没啥意义。反之，访问延迟很短，但是吞吐率很低，同样没有意义。

所以，一个系统的性能必然受到这两个条件的同时作用。

资源使用率

一个系统和服务总是需要软硬件容量来支撑的，那么**资源的使用率**就至关重要了。因为它直接决定了系统和服务的运营成本。

这一指标虽然主要是面向系统容量的，但其实和客户也直接相关。如果资源使用率低，比如 CPU 使用率低，在系统容量固定（比如服务器数目固定）的情况下，吞吐率也会较低，或

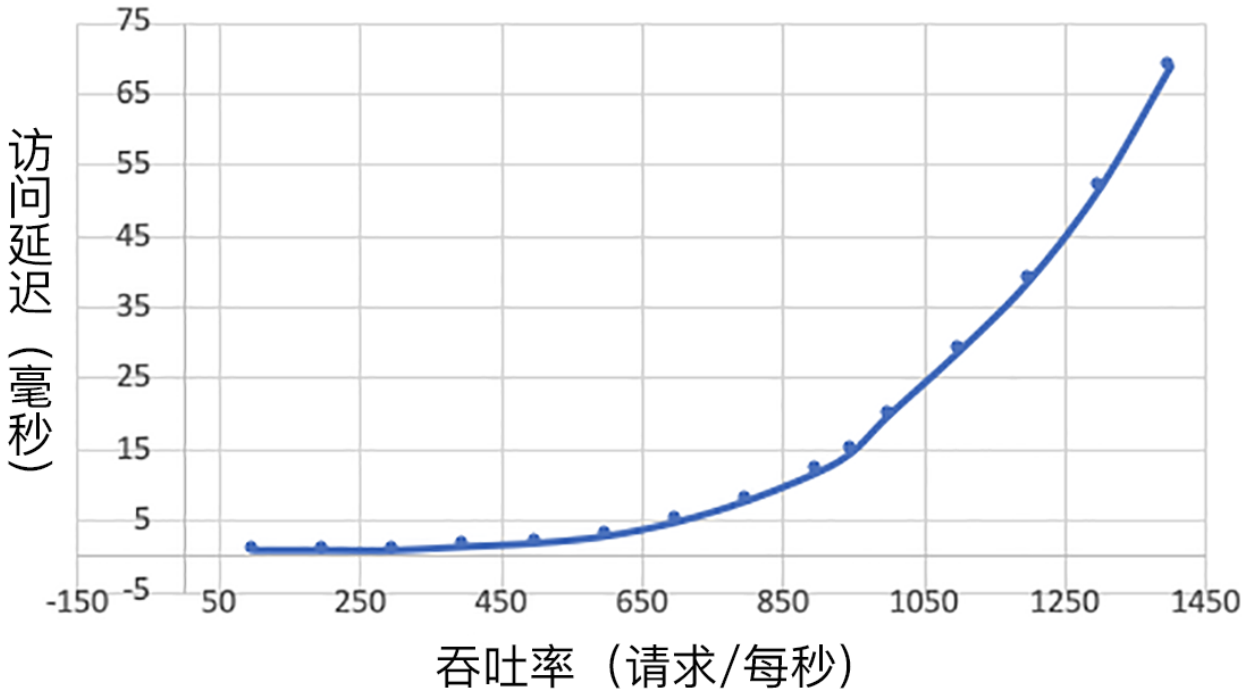
者访问延迟会较高，因为系统资源没有被充分利用。

外部性能指标的变化

我们还需要知道，这三个性能指标的变化有它们自己的特点，而且经常会互相影响。

对一个系统而言，如果吞吐率很低，**服务延迟**往往会非常稳定。当吞吐率增高时，访问延迟一般会快速增加。

下图展示了一个有代表性的系统的吞吐率和访问延迟的变化。



在吞吐率低于每秒 600 时，访问延迟小于 5 毫秒，这个延迟相对稳定。然后随着吞吐率变大，访问延迟飞速攀升。一般而言，根据系统的延迟可接受大小，我们需要控制负载流量，以免访问延迟过大而影响客户体验。

我们在测量访问延迟的时候，不仅要计算均值，还需要注意延迟的分布情况，比如，有百分之几的在服务允许的范围，有百分之几的略微超出了，有百分之几的完全不可接受。多数情况下，平均延迟达标了，但是其中可能有很大比例（比如 20%）远远超出了我们可接受的范围。

所以，我们在规定延迟标准的时候，除了均值，还需要定义百分位数的可接受值。比如，平均延迟 10 毫秒，P90 小于 30 毫秒，P99 小于 50 毫秒等等。

关于**吞吐率**，现实中的系统往往有一个峰值极限。

超过这个峰值极限，系统就会超载，除了服务延迟超标，还会造成一系列的性能问题（比如系统挂掉）。这个峰值极限往往需要经过仔细的性能测试，并且结合访问延迟标准来确定。有了这个峰值极限值后，系统的设计和运维就需要确保系统的负载不要超过这个值。

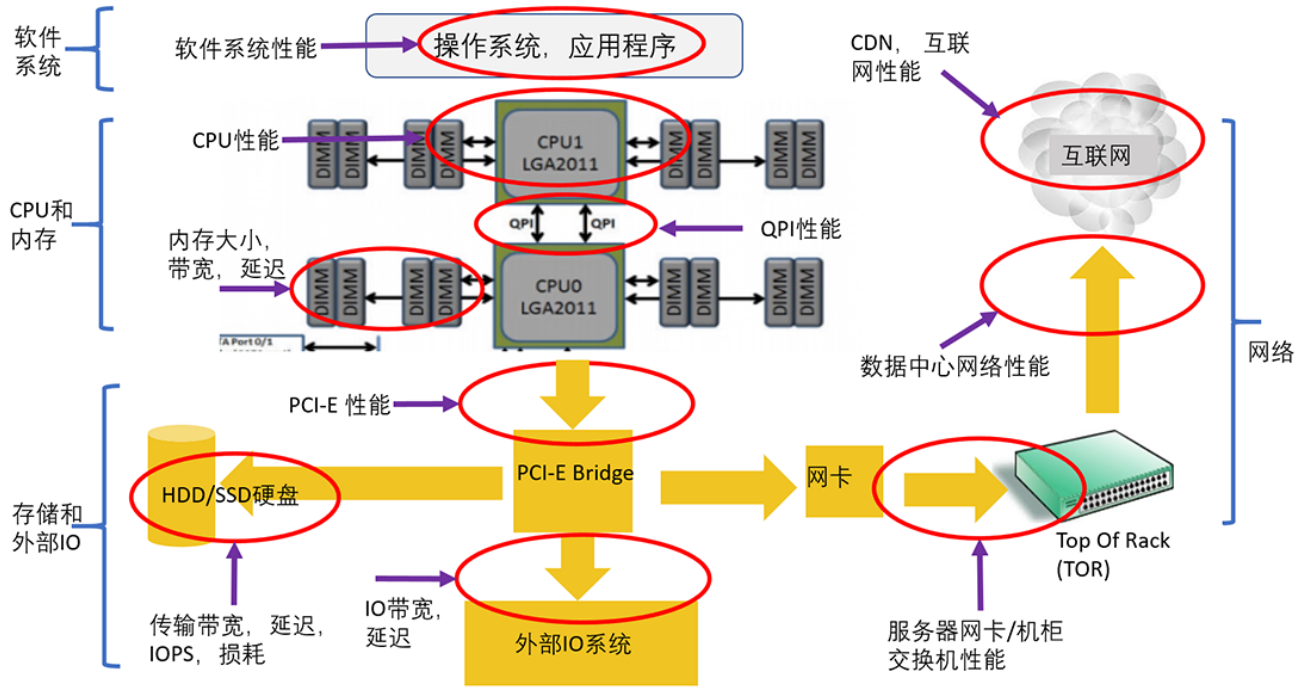
除了影响运营成本和系统容量，**资源使用率**的标准也需要考虑其他几个重要因素。

一个因素是意外事件的缓冲（Buffer）和灾难恢复（Disaster Recovery, or DR）。一个现实世界中的系统，随时都会有意外事件（比如流量波动）或者部分网络故障，这就需要整个系统资源保留一定的缓冲，来应付这些意外和从发生的灾难中恢复。比如 CPU 的使用率，虽然理论上可以到 100%，但考虑这些因素，实际的使用率指标往往远远低于 100%。

性能问题归根结底是某个资源不够

所有的性能问题，虽然表现方式各异，归根结底都是某种资源受到制约，不够用了。这里的资源指的是一个计算机系统，程序和互联网服务会用到的每一种资源，比如 CPU、网络等。换句话说，客户的请求在处理时在某个地方“卡住了”。这个卡住的地方就叫“瓶颈”（或者叫卡点，Choke point）。

根据我的经验，我在下面这张图表中展示了一个系统常见的十大瓶颈，基本上覆盖了所有可能出现性能问题的地方。



这十大瓶颈可以大致分为四类，在后面的几讲中我们会详细讨论分析每一类别里面的具体性能问题。这四类是：

1. 软件系统：包括操作系统、应用程序、各种类库以及文件系统。
2. CPU 和内存：包括 CPU 性能、QPI（QuickPath Interconnect，处理器之间的快速通道互联）和缓存内存。
3. 存储和外部 IO：包括处理器的 IO 的接口性能、各种存储系统（尤其是 HDD 和 SSD 性能）。
4. 网络：包括服务器到机柜交换机的网络、数据中心的网络、CDN 和互联网。

总体上来讲，性能分析的目的，是提供高性能、低延迟、高效率的服务。

要实现这一目的，就需要找到系统和服务的性能瓶颈，然后尽可能的消除瓶颈，或者降低瓶颈带来的影响。系统和服务有性能瓶颈就说明这个地方的资源不够用了。所谓最大的性能瓶颈，就是说这个地方的资源短缺程度最大，相对而言，其他地方的资源有富余。

如何找到最大的性能瓶颈？

这就需要进行性能测试和性能分析了。性能分析时需要知道三个层次的知识：

第一个层次是可能的性能瓶颈，比如我们刚刚讨论的十大瓶颈。知道了瓶颈才能有目标的去分析。

第二个层次是每个瓶颈有哪些资源有可能短缺。比如内存就有很多种不同的资源，不仅仅是简单的内存大小。除了内存使用量，还有内存带宽和内存访问延迟。

第三个层次是对每个瓶颈的每种资源要了解它和其他模块是如何交互的，对整个系统性能是如何影响的，它的正常值和极限值是多少，如何分析测量等等。

找到性能最大瓶颈后，具体的优化方式就是什么资源不够就加什么资源，同时尽量降低资源消耗，这样就可以做到在资源总量一定的情况下，有能力支撑更高的吞吐率和实现更低的延迟。

依据数据和剖析（Profiling）来分析

做性能分析时，必须采用科学的方法，尽量依据数据，来引导我们的分析和验证我们的推论，而不是完全凭空猜测。

当我们有了比较多的性能分析和优化的经验后，慢慢就会对一个系统的内部各个模块的交互，以及各种性能问题肚里有数了。这种时候，适度地做一些理论推测是合理的。就像一个有经验的医生，往往稍微了解一下病人的情况，就猜个八九不离十。这就是经验的重要性。

不过，再有经验的医生，还是需要做进一步的检验，尤其是面对复杂的病人和病情。同样的，无论性能分析的经验多丰富，我们也需要谨慎地做性能测试和数据分析，尤其是在针对重要系统的时候。

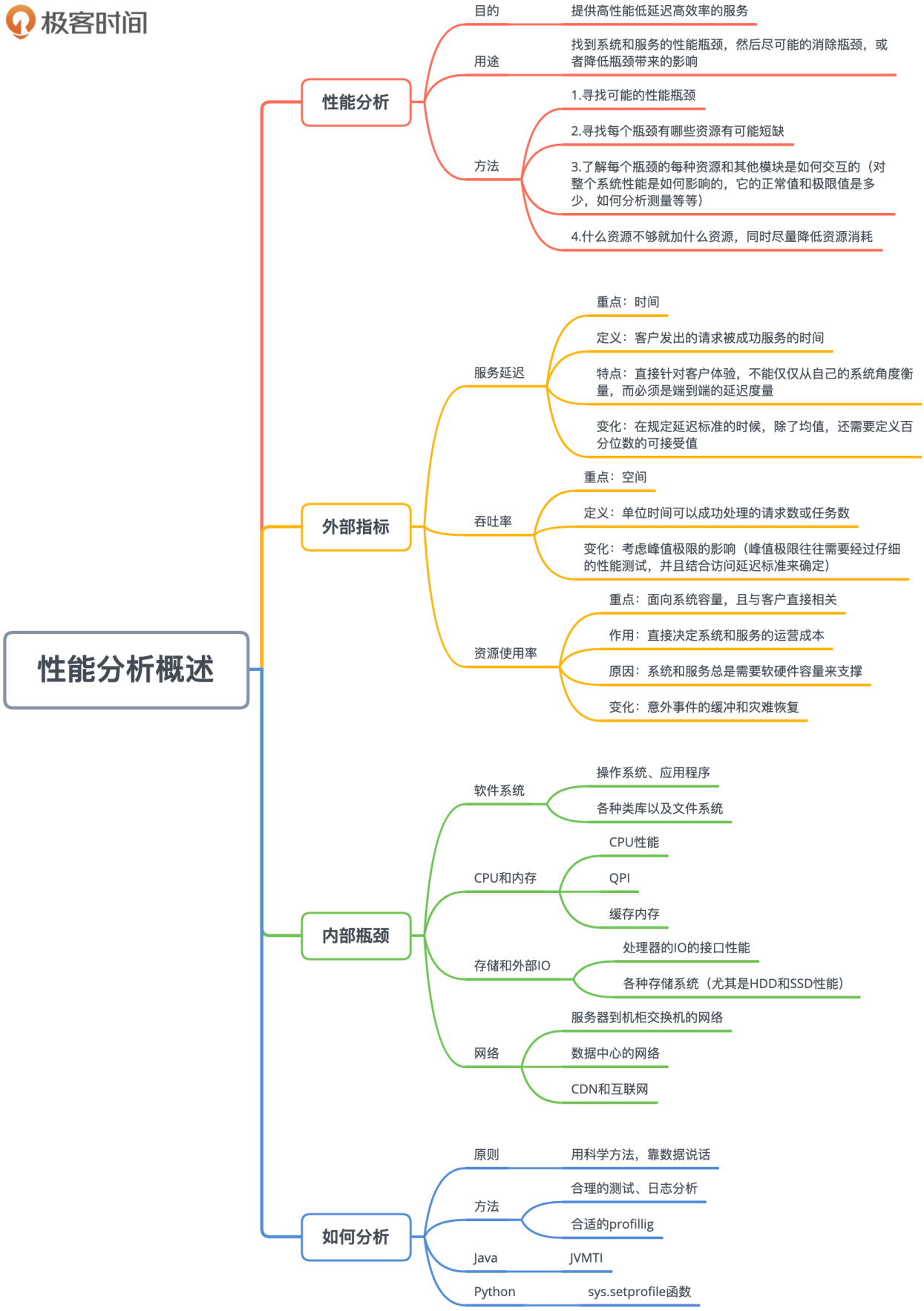
这一点可以说是性能分析和优化的第一原则。当我们怀疑性能有问题的时候，应该通过合理的测试、日志分析，并作合适的剖析（Profillig），来分析出哪里有问题，从而有的放矢，而不是凭感觉、撞运气。

比如，如果是 CPU 相关的性能问题。按照我们学过的帕累托 80/20 定律，系统绝大多数的时间应该都耗费在少量的代码片段里面。如何找出这些需要优化的代码呢？唯一可靠的办法就是 profile。现代流行的各种编程语言，比如 Java 和 Python 等，都有相关的 profile 工具。所以，会熟练使用这些 profile 工具是性能分析和优化的必要条件。

我们举几个例子，比如 Java 语言，就有很多工具，像 JVMTI（JVM Tools Interface，JVM 工具接口），就为性能分析器提供了方便的钩子，可以用来跟踪诸如函数调用、线程相关的事件、类加载之类的事件。再比如对 Python 语言来说，我们可以用 sys.setprofile 函数，跟踪 Python 的函数调用返回异常等事件。

总结

我们的儒家思想提倡“格物致知”，就是说要深入探究事物的原理，而从中获得知识和智慧。性能优化能否成功，也需要探究一个系统中性能的真正问题，找到性能的最大瓶颈。



格物致知时，还需要“正心“和”诚意”，就是要实事求是和端正态度，从而科学而系统地收获知识。我们做性能分析时候，也是需要根据实际的数据和 Profiling 等测试结果，找到

性能的瓶颈，并合理地解决性能问题。

思考题

你工作中有没有碰到没有搞明白的性能问题？如果有，想想能不能按照今天讲的几个可能的性能问题领域来一个个考虑并验证一下？说不定会有“守得云开见月明”的恍然大悟呢。

欢迎你在留言区分享自己的思考，与我和其他同学一起讨论，也欢迎你把文章分享给自己的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　13 | 性能测试的工程集成：如何与产品开发和运维业务有机集成?

下一篇　15 | 常见的性能问题之CPU篇：如何让CPU的运行不受阻碍?

精选留言 (1)

写留言

王治澎
2019-12-29

配上思维导图，整个知识树越来越高

展开